

OBJECT TÜRÜ

▼ Object nedir?

- tüm türleri (değer türü, referans türü..)karşılayabilen bir **türdür**.
- Tüm türler varsayılan olarak object'ten türerler.
- Referans türü bir değişkendir ama değer türü değerleri de karşılayabilir.



▼ Boxing

- Object türdeki bir değişkene herhangi bir türdeki değeri göndermek boxing olarak nitelendirilmektedir.

using System;

```

namespace objectType
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            #region object
            #region Boxing
            //object türdeki bir değışkene herhangi bir türdeki değeri göndermek Boxing olarak nitelendirilmektedir.

            int yas = 28;
            object _yas = 28; //Boxing

            #endregion
            Cast Operatörü
            UnBoxing
            #endregion
        }
    }
}

```

Herhangi bir deęer object türüne assign ediliyorsa eęer bu işlem Boxing'dir...

Boxing işlemi neticesinde ilgili deęer objectin içerisinde kendi türüyle saklanır. Fakat _yas deęiskeni artık 28 deęerini bizlere object türde getirecektir.

Burada dikkat ederseniz object türden elde edilen deęer üzerinde turuna özgü işlemler gerçekleştirilemez! Örneğin; sayısal bir deęer varsa o deęer object olarak geleceğinden dolayı matematik işlemleri yapılamaz!

Object bir deęiskenin içerisindeki deęer üzerinde turuna özgü işlemler yapılabilmek için o object'in içerisinde deęeri kendi/has/özgen türünde elde etmemiz gerekmektedir. İste bu işlemde UnBoxing diyeceğiz...

object_yas
(int)28
_yas → (object)28

Boxing

▼ Cast operatörü

- Boxing edilmiş bir veriyi (object'e atanmış veriyi) kendi türünde elde etmemizi sağlayan bir operatördür.
- Tür dönüşümlerinin bilinçli tür dönüşümü konusunda Cast operatörünü kullanacağız.
- Kalıtsal durumlarda da karşılaşacağız..
- () → cast operatörü parantezdir.
- Cast operatörü, object olan deęişkenin solunda o object'in hangi türe Unboxing etmek istiyorsak parantezin içerisinde hedef türü bildirerek kullanılır.

Cast Operatörü

Boxing edilmiş bir veriyi kendi türünde elde etmemizi sağlayan bir operatördür.

Tür dönüşümlerinde bilinli tür dönüşümü konusunda Cast operatörünü kullanacağız.

Kalıtsal durumlar da da karşınıza çıkacaktır.

Cast operatörü parantez'dir.
Cast operatörü, object olan deęiskenin solunda o object'i hangi türe Unboxing etmek istiyorsak parantez içerisinde hedef türü bildirerek kullanılır.

(T)object;

int a = 5;
object b = a;

Boxing

(int)b;

Cast operatörü b deęiskeni/objesi içerisindeki deęeri bana int olarak ver demektedir.

▼ Unboxing

```
9
10 #region object
11 #region Boxing
12 //object türdeki bir değişkene herhangi bir türdeki değeri göndermek Boxing olarak nitelendirilmektedir.
13
14 object _yas = 28; //Boxing
15
16 #endregion
17 Cast Operatörü
18 #region UnBoxing
19
20 #region Casting
21 int yas = (int)_yas; //UnBoxing
22 Console.WriteLine(yas * 5);
23
24 #endregion
25 #endregion
26 #endregion
27
28
29
30
31
```

UnBoxing

object_yas

Boxing

(int) 28

UnBoxing esnasında Boxing edilmiş verinin orijinal türü ne ise o bildirilmelidir. Yani int türünde boxing edilmiş bir değeri int türünde UnBoxing etmelisiniz... Gidip int türünde boxing edilmiş bir veriyi char türünde UnBoxing etmeye çalışırsanız

▼ var keyword'ü

var Keyword'u

Tutulacak değerin türüne uygun bir değişken tanımlayabilmek için kullanılan keyword'dür. var keywordü, kendisine atnaan değerin türüne burunur.

var keywordü, compiler tarafından değerin türüne göre otomatik burundurulun bir keyworddür. Lakin bir tür DEĞİLDİR!

Genellikle, yazacağımız deiskenlerin türlerini yazmaktan usendığımız için kullanmaktayız!!! Halbuki esas olma sebebi ise farklı diller arasında desteklenmeyen türlerdeki verileri karşılayabilmek için oluşturulmuş ortak bir keyworddür.

Diller arasındaki entegrasyonda kullanılıyordu...

bool medeniHal = true;

var medeniHal = true;

(bool olduğundan dolayı var bool türüne burunecektir)

var yas = 28;

Türünü bilebildiğimiz verilerin değisken türlerini var ile compiler'a yaptırmak ufakta olsa bir maliyettir! Bunun için bizler bu maliyet kaçınmak adına bildiğimiz türleri mümkün mertebe usenmeden belirleteceğiz...

**** Eğitim sürecinde bazen usendığım için bende var kullanacağım :)))) ****

```
#region var
string adi = "Gencay";

var x = 3.14;

//1. var keywordüyle tanımlanan değiskenin değeri tanımlanma aşamasında verilmelidir. Verilmelidir ki türü belirleyip
direkt ona dönüşebilsin o türde davranış sergileyebilsin...
//2. var keywordüyle tanımlanan değiskenin ilk değeri verildikten sonra o değerin türüne bürüneceği için sonraki
durumlarda değeri farklı türlerde verilemez!!!
//3. var - object arasındaki farkı ;
//var bir keyword iken object ise bir türdür. var atanan değerin türüne bürünürken, object atanan değeri Boxing
yaparak object'te dönüştürür.
#endregion
```

▼ Dynamic keyword'ü

Var tanımlanır tanımlanmaz, verinin türünü gösterir. Dinamikte ise bu böyle değildir, veriyi uzaktan alırız, uygulamayı build etmeden önce verinin bize hangi türde gönderildiğini anlayamayız. Runtime sırasında meydana çıkır ne olduğu.



```
0 references
static void Main(string[] args)
{
    #region dynamic
    //var a = 5;
    //dynamic _a = "5";
    //_a.ToStringAhmet();
    //Console.WriteLine(_a.GetType());
    //Console.WriteLine(_a * 5);

    //dynamic keywordü runtime'da türü belirleyecektir lakin kararlı davranmayacaktır.
    dynamic x = "Ahmet";
    Console.WriteLine(x.GetType());
    x = 3.14D;
    Console.WriteLine(x.GetType());

    #endregion
}

Microsoft Visual Studio Debug Console
System.String
System.Double
C:\Users\Gençay\Source\Repos\Application\var_dynamic_keywords\bin\Debug\netcoreapp3.
8276) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debu
le when debugging stops.
Press any key to close this window . . .
```

Dynamic Nereelerde tercih edilir???



int, bool, Personel, İnsan, Satis v.s .vs.

data

dynamic

yyyyyy....

Uzaktan gelen veriler var keywordu ile karsilanamaz!!!

Çunku var keywordu tanimlandigi esnada verinin atanmasini ister!!!